

AN FPGA BASED TESTBENCH FOR RELIABILITY AND ENDURANCE
CHARACTERIZATION OF NONVOLATILE MEMORY

BY

VIKRAM M. N. RAO

B.S., University of Alaska, Fairbanks, 1999

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2002

Urbana, Illinois

ABSTRACT

This research describes a hardware test platform and test methodology that was developed for NASA's Electronic Parts and Packaging Program. Nonvolatile memories will play an increasingly important role in future NASA missions. Autonomous systems require nonvolatile memories to store the system's states, programs, and data, and the limited bandwidth of communication during distant missions also makes reliable nonvolatile memory storage a necessity. A test bench has been developed for testing memory chips from various nonvolatile memory technologies, using a Xilinx XC4010E (10,000 gate) FPGA. The test bench is being used to perform endurance, reliability, and MINVDD testing. A MATS+ memory test, which can detect address decoder faults and stuck-at faults and cycles through all the addresses in the memory, was chosen for reliability testing. For endurance testing, specific data patterns are written to and read from the same address range of the memory continuously, which allows for faster endurance testing, especially when slower memory technologies are used. In MINVDD testing, the minimum supply voltage at which a chip can function correctly is determined. This information is then used to screen out "weak" chips. In all of these tests, the data logged upon each error includes the error number, the address at which the error occurred, the cycle number (where one cycle is defined as one read or write operation to a single address), the incorrect data value read, and (for the MATS+ test) the portion of the test in which the error occurred. This test bench offers several advantages over commercial testers when used for reliability and endurance testing. Endurance testing to a chip's specifications could involve more than 10^{10} read/write cycles, which can take up to 28 days for the Ramtron FM24C04 serial FRAM. Commercially available memory testers with high hourly rates may prove extremely expensive

for testing nonvolatile memories with 10^{12} to 10^{15} read/write cycles. In comparison, the FPGA-based testers are inexpensive and more flexible. If several FPGA boards are used, many chips can be tested simultaneously at a fraction of the cost compared to the commercial testers. No errors have been found in reliability, endurance, or MINVDD testing on any of the memories tested.

To my family

ACKNOWLEDGMENTS

This work was supported by the NASA Jet Propulsion Laboratory, contract no. 1229911. I would like to thank my advisor, Professor Janak H. Patel, for many insightful discussions, helpful suggestions, and encouragement that made this work possible. Working with Professor Patel has made graduate research a rewarding experience, and I feel lucky to have had the opportunity to work with a professor with such a history of groundbreaking research. I would also like to thank Dr. Jagdish Patel, of the Jet Propulsion Laboratory, for his help and for the use of the Jet Propulsion Laboratory facilities and equipment to facilitate this project.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1 Background.....	1
1.2 Memory Technology.....	2
1.3 FRAM Operation.....	2
1.4 EEPROM Operation.....	6
2 RELATED WORK	9
3 RELIABILITY ISSUES	13
3.1 Reliability Issues of FRAM.....	13
3.2 Reliability Issues of EEPROM.....	14
4 FAULT MODELS	17
4.1 Stuck-At Fault.....	18
4.2 Transition Fault.....	19
4.3 Coupling Fault.....	19
4.4 Neighborhood Pattern-Sensitive Fault.....	20
5 TESTER DESIGN	23
5.1 Design Choices and Design Flow.....	23
5.2 Memory-Specific Tester Design Issues.....	25
5.3 Tester Operation.....	28
5.4 Tester Design Modification.....	29
6 TEST METHODOLOGY	30
6.1 Reliability Testing.....	30
6.2 Endurance Testing.....	31
6.3 MINVDD Testing.....	32
7 EXPERIMENTAL RESULTS	33
8 CONCLUSIONS AND FUTURE WORK	35
REFERENCES	37
APPENDIX A. NVM TESTER USER'S GUIDE	40
APPENDIX B. MEMORY TESTER CIRCUIT SCHEMATICS	43
APPENDIX C. VERILOG CODE SAMPLE (MATS+ reliability testing)	46
APPENDIX D. VERILOG CODE AND USER CONSTRAINT FILES (CD-ROM)	64

1 INTRODUCTION

This research describes a hardware test bench and test methodology that was developed for NASA's Electronic Parts and Packaging Program. This work was supported by the NASA Jet Propulsion Laboratory (JPL).

1.1 Background

Future NASA missions will demand higher performance and functionality from computer systems on long duration missions. This will involve autonomous systems that require very large, reliable nonvolatile memory systems to store the system's states, programs, and data. In addition, the low communication bandwidth inherent to deep space missions makes reliable nonvolatile memory storage a necessity. Commercially available CMOS based nonvolatile memory systems suffer from single-event upset (SEU) problems because the information is stored in the form of a small quantity of charge, and because of the radiation-induced total dose effects caused by the degradation of oxide layers actively involved in the functioning of memory cells [1]. All nonvolatile memories suffer from endurance limitations, so that only a limited number of read and write operations can be performed before the data retention and reliability of the memory are compromised. In addition to limited endurance, EEPROM memories suffer from slow programming times. Manufacturers of ferroelectric semiconductor memories, which have recently been qualified for commercial production, claim to have minimized these problems. With high radiation resistance, high endurance ratings, excellent retention over a wide range of temperatures, and fast programming times, ferroelectric memories provide significant advantages over traditional nonvolatile memories [2]. However,

due to the relatively recent introduction of ferroelectric memory in the commercial market, there exists limited data on the reliability and endurance characteristics of FRAM. The research described in this paper is part of an NEPP project, in which a hardware test platform and test methodology have been developed to determine reliability and endurance characteristics of certain nonvolatile memories.

1.2 Memory Technology

Memory technologies are divided into two categories. The first category, nonvolatile memories, are characterized by their ability to retain data in the absence of power. These memories are traditionally used in read-only or read-mostly applications because of their limited write endurance and slow write speed. Nonvolatile memories are derivatives of ROM technology, which includes EPROM, EEPROM, flash, and more recently ferroelectric nonvolatile memory technology. The second category of memory technology, volatile memories, are RAM-based devices including SRAM and DRAM. Writing to these memories is fast and write endurance is unlimited, so they are most often used to store data that change frequently, but they cannot store data in the absence of power. Nonvolatile memory technologies with promising future potential include FRAM, Chalcogenide RAM, GMRAM, Tunneling MRAM, and SONOS EEPROM.

1.3 FRAM Operation

FRAM is a RAM-based memory that uses the ferroelectric effect as its storage mechanism. The ferroelectric effect is the ability of a material to store a state of electric polarization in the absence of an applied electric field [3]. This is a very different mechanism

from that used in conventional nonvolatile memories, which use floating gate technology. A FRAM memory cell is created by depositing a film of ferroelectric material in crystal form between two electrode plates to form a capacitor very similar to a DRAM capacitor. Current FRAM memory cell designs utilize one transistor one capacitor (1T1C) or a more robust design using two transistors two capacitors (2T2C) for better fault tolerance (complementary storing). However, rather than storing data as a charge on the capacitor like DRAM, a ferroelectric memory stores data within a crystalline structure known as Perovskite. The Perovskite crystals maintain two stable polarization states resulting from the alignment of internal dipoles, which are used to represent '1' and '0' states. Figure 1 [4] shows the hysteresis curve for a ferroelectric capacitor and the response of the remnant polarization (P_r) and nonremnant (spontaneous) polarization (P_s) to the externally applied electric field (E) [5]. The hysteresis curve saturates at P_{sat} , where the maximum alignment of spontaneous polarization occurs. After reaching P_{sat} , when the electric field is removed instantaneously, the electronic polarization associated with the linear capacitance component decreases to zero, while the spontaneous polarization (P_s) remains. Then, usually within a few milliseconds, the polarization decays to the remnant polarization (P_r). The separation along the y axis at 0 V is a measure of the remnant polarization for the capacitor.

A simplified model of a ferroelectric crystal is shown in Figure 2 [3]. It has a mobile atom in the center of the crystal. Applying an electric field across the face of the crystal causes this atom to move in the direction of the field, and reversing the field causes the atom to move in the opposite direction. Since no external electric field is required for the ferroelectric material to remain its polarization, a ferroelectric memory device can retain data in the absence of power and requires no periodic refresh.

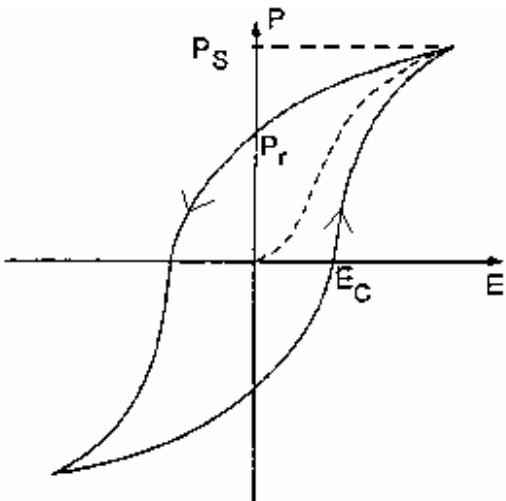


Figure 1 Hysteresis Curve for a Ferroelectric Capacitor [4]

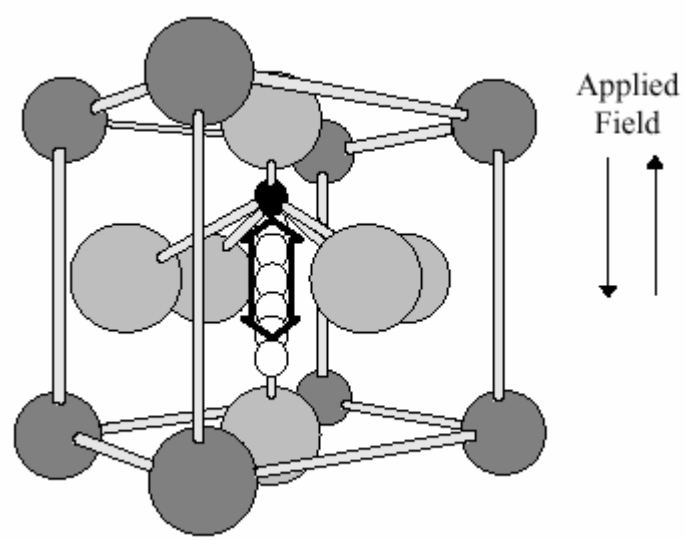


Figure 2 Simple Model of a Ferroelectric Crystal [3]

Although the memory element of an FRAM cell is a capacitor, it does not store data as a charge. In order to read a FRAM memory cell, it is necessary to detect the position of the atoms within the Perovskite crystals. However, this cannot be directly sensed. Instead, an electric field is applied across the capacitor. The mobile atoms move across the crystals in the direction of the field if they are not already in the appropriate positions. The circuit dumps charge resulting

from the applied field from the capacitor and compares it to the charge from a reference. A capacitor with atoms that switch states will emit a larger charge than a capacitor with atoms that do not switch states. Sense amplifiers built into the FRAM chips measure this charge and produce either a zero or one on the output pins. Since a memory read operation involves a change of state (it is a destructive read), the circuit automatically restores the memory state. Therefore, each read access is accompanied by a precharge operation that restores the memory state.

A write operation is very similar to a read operation. The circuit applies data to be written to the ferroelectric capacitors. If necessary, the new data simply switches the state of the ferroelectric crystals. As with a read, a precharge operation follows a write operation.

Current FRAM products use a two-transistor, two-capacitor (2T2C) memory cell, which provides an individual reference in close proximity for each data bit. Depending on the programmed data state, one capacitor will switch when read while the other will not switch. The assignment of '1' and '0' states is arbitrary during the memory design. Given the close proximity of the reference to the data bit, the memory circuit can measure the charge difference between the switching and nonswitching capacitors very precisely. Variations in the capacitors across the memory array are eliminated from consideration by having a differential signal for each bit. An example of a 2T2C memory cell is shown in Figure 3. More advanced one-transistor, one-capacitor cell technology, in which data values are compared to a global reference, have recently become commercially available.

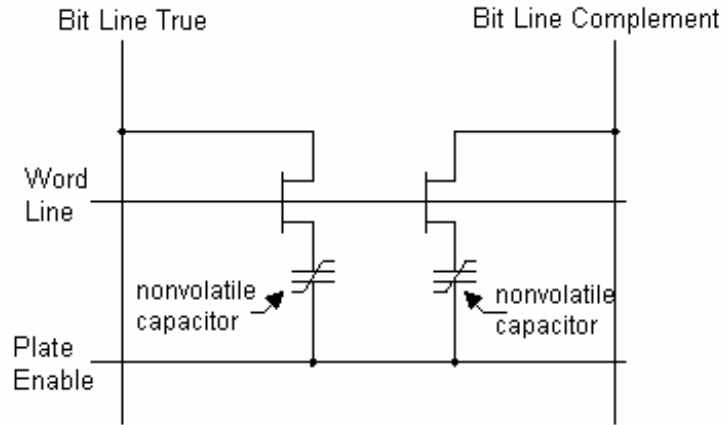


Figure 3 A 2T2C Ferroelectric Memory Cell [3]

1.4 EEPROM Operation

The two dominant EEPROM (electrically erasable programmable read-only memory) technologies include FLOTOX (floating gate tunnel oxide), and MNOS (metal nitride oxide silicon), the latter also including silicon-oxide-nitride-oxide-semiconductor (SONOS) technology [6]. In FLOTOX-based EEPROMs, program and erase operations are carried out by electron tunneling through an oxide. SONOS technology, used in EEPROM memories tested in this research, will be described in further detail.

EEPROMs based on floating gate technology use memory cells with transistors that are very similar to normal MOS transistors, but the transistors have a second, floating gate. Applying a programming voltage V_{PP} (usually greater than 12 V) to the drain of the n-channel EEPROM transistor programs the EEPROM cell. A high electric field causes electrons flowing toward the drain to move so fast they “jump” across the insulating gate oxide where they are trapped on the bottom, floating, gate. The energetic electrons are referred to as hot and the effect is known as hot-electron injection or avalanche injection. Electrons trapped on the floating gate raise the threshold voltage of the n-channel EEPROM. Once programmed, an n-

channel EEPROM device remains off even with a logic high applied to the top gate. An unprogrammed n-channel device will turn on as normal with a logic high top-gate voltage. The programming voltage is applied either from a special programming box or by using on-chip charge pumps. In programming an EEPROM, an electric field is used to remove electrons from the floating gate of a programmed transistor.

SONOS is an acronym for silicon-oxide-nitride-oxide-silicon. The memory device is a silicon gate N-channel MOS transistor with a specially processed gate dielectric consisting of a tunneling oxide, a silicon nitride layer, and a capping oxide. The SONOS memory effect relies on charge storage within the silicon nitride film, with the silicon dioxide above and below it acting as energy barriers to the loss of charge [7]. The charge is injected by tunneling through the tunneling oxide. The charge deposited in the SONOS dielectric does decay slowly with time, but when written under the specified conditions and stored within the specified limits, data is indeed permanent for most purposes. While modern flash EEPROMs use a single floating gate transistor per bit and data detection is sensed by a single ended reference scheme, the SONOS EEPROMs used for testing were radiation hardened, and used a much less dense but more robust differential sensing mechanism. Two variable threshold SONOS transistors and two fixed threshold devices make up each bit. The two devices per bit are always programmed in opposite threshold voltage states in a two-step process of erasure to the negative voltage threshold state followed by writing of one of the two devices to the positive threshold voltage state. Due to this differential sensing approach, erasure results in an indeterminate logic state and is always followed by a write of one of the two SONOS transistors to the positive threshold voltage state. Common to most EEPROMs, an entire page is written at a time and byte-erasure is not supported. Figure 4 [8] shows the write/erase

operation of a SONOS device, which has a multilayer dielectric gate insulator consisting of an oxide-nitride-oxide sandwich with charge storage in discrete traps in the silicon nitride layer. A net positive or negative charge is stored in deep traps within the nitride dielectric depending on whether a positive or negative voltage is applied, respectively, to the gate electrode. For illustration purposes we assume a +10 V programming voltage is applied on the gate relative to the p-substrate, which forms an electron channel or inversion layer. During a program operation (write), electrons quantum-mechanically tunnel from the silicon inversion layer through an energy barrier height of 3.1 eV into an ultrathin oxide into the silicon nitride film and are stored in deep-level traps, which lie about 1 eV below the edge of the nitride conduction band.

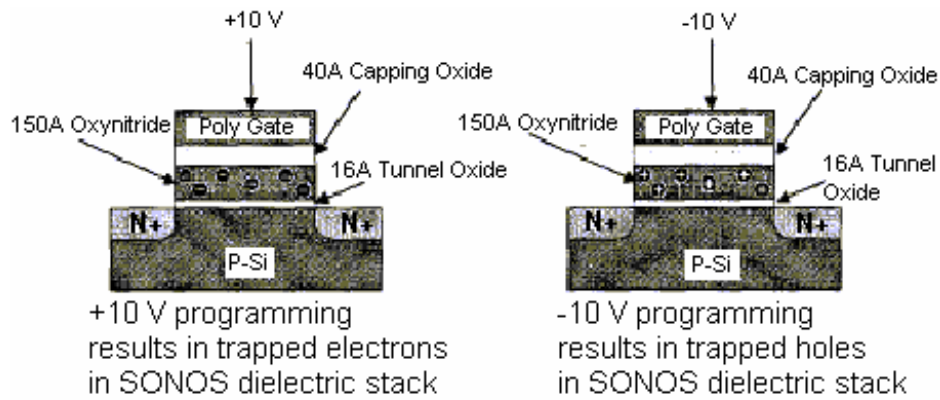


Figure 4 SONOS EEPROM Programming Mechanism [8]

2 RELATED WORK

Determining the suitability of nonvolatile memories for space applications is a subject of ongoing research. Over the last several years, many approaches have been developed to characterize the reliability and endurance of nonvolatile memories. Studies involving ferroelectric memories are described below:

- An experimental procedure used in the evaluation of data retention after fatigue on ferroelectric memories is described in [9]. An experimental procedure was developed that would first fatigue a FRAM memory at elevated voltages, and then perform a long-term retention/imprint test. The Ramtron FM1608 64 kb parallel FRAM was chosen for testing. Using a burn-in board with 105 FRAMs per board, the chips were fatigued at an elevated voltage, citing previous work [10] which has shown that, for a given ferroelectric thickness, increasing the potential across the material brings the onset of degradation at a lower number of fatigue cycles. The chips were fatigued to $1E10$ fatigue cycles at elevated voltages of 5-7 V. This was followed by a full functional test, then a data retention bake with all data bits set to 0 at a temperature of 200 °C for 72 hours, and finally a 70 °C bake for 10 min in the complemented data state (all data bits set to 1). Previous work [11, 12] has reported that the retention performance of ferroelectric memories follows an Arrhenius behavior with an activation energy of 0.87 eV. This was used to calculate the equivalent aging caused by the data retention bake, which was equivalent to 100 years at 55 °C or 7.75 years at 85 °C. The failure fraction was calculated as the ratio of failed bits to bits tested, and the failure fraction was extrapolated for a range of voltages and fatigue cycles.

- Similarly, [13] describes Ramtron's test methodology used in testing the FM24C16 FRAM. An accelerated retention study was conducted by writing the FRAMs with a pattern, baking the chips, and then reading the pattern. Median lifetimes and activation energies were calculated.
- In [14], 100 Ramtron FM1608 64 kb FRAMs were divided into various groups and subjected to various data retention and endurance tests. Electrical measurements included parametric and functional measurements. Data retention tests were conducted after performing high-temperature storage aging, low-temperature exposure, temperature cycling, read-write cycling, and total-dose ionizing radiation exposure. Fatigue testing consisted of subjecting the parts to $3E7$ write-read cycles while taking parametric and functional measurements. It should be noted that the endurance specification for the Ramtron FM1608 is $1E10$ write-read cycles.
- In [15], the effects of temperature, electric field, and the number of polarization reversals on ferroelectric memory aging were analyzed. Although the experiment did not test an actual FRAM, ferroelectric capacitors were tested in an environment similar to that of a memory design. A relationship was found between the number of read/write cycles a capacitor has been exposed to and its retention lifetime. Signal loss was found to degrade linearly with the log of time.
- In [2] and [16], standard process qualification testing procedures for SBT-based ferroelectric memories are described. High-temperature storage, thermal shock, and pressure cooker test results show that ferroelectric memories based on SBT technology have reliability levels comparable to other semiconductor memories. The retention performance for the ferroelectric manufacturing process is related to operating

temperature and a method is described to calculate the FIT failure rate based on temperature. Radiation experiments for total ionizing dose, neutrons, heavy ion, and proton exposure show the ferroelectric storage element to be radiation tolerant. The limiting factor in radiation hardness was found to be the radiation hardness of the underlying CMOS circuitry.

- In [13], the endurance properties of ferroelectric PZT thin films are examined. The polarization characteristics of these films were determined from large-signal quasi-static capacitance-voltage and high-frequency, small-signal, capacitance-voltage measurements. It was found that a trade-off exists between expected storage capacity and endurance. High operating fields increase the initial storage capacity but accelerate fatigue, whereas low-field operation can improve endurance but has limited storage capacity. The results are deemed to be consistent with a described domain-pinning model.

Prior research has also examined the reliability and endurance characteristics of various EEPROM technologies. Previous work in this area includes:

- In [17], various endurance, data retention, and disturb failure modes are described for floating-gate-based EEPROMs. These mechanisms include stuck bit faults, retention degradation, read time degradation, erase time degradation, program time degradation, disturbs, overerase, erase disturbs, program disturbs, and read disturbs. A test methodology is presented as part of a comprehensive reliability verification program following JEDEC standard tests.
- In [18], it is shown that the endurance of a floating-gate-based EEPROM can be improved by decreasing the electric field across the tunnel oxide with an appropriate

programming signal. Endurance testing shows that an optimized programming signal can reduce the duration of the stress induced by the electric field across the tunnel oxide, thereby decreasing the rate of degradation of the oxide.

- In [19], common test procedures used in reliability and endurance testing of EEPROMs are described. Endurance failure mechanisms are presented for floating-gate-based EEPROMs. The effect of endurance cycling on data retention performance is found to be slight.
- In [20], SONOS-based EEPROMs are tested for use in space and military applications. Data retention tests indicate SONOS nonvolatile memories with 10^6 to 10^7 erase-write cycles are possible. The radiation hardness of an NGC 64k EEPROM was found to be over 300 krads.

All of the above studies extrapolate endurance characteristics from accelerated heat testing. The research described in this paper differs in that real-world endurance results are presented, and a new test bench has been developed for each nonvolatile memory tested.

While the storage material differs in the FRAM and EEPROM memories tested, the address decoder and sense amplifier circuitry are still primarily CMOS-based. Therefore, MINVDD test techniques were applied to detect weak CMOS circuits. This test method is relatively recent, and was presented in the following previous work:

- In [21], the recently introduced MINVDD test technique, used to detect weak CMOS ICs, is presented. The minvdd of a chip is defined as the minimum supply voltage at which a chip can function correctly. The minvdd values for weak chips containing various types of flaws, including metal shorts, gate oxide shorts, threshold voltage shifts, and tunneling opens are determined.

3 RELIABILITY ISSUES

Although data retention and endurance characteristics are limiting factors of both FRAMs and EEPROMs, the degradation mechanisms by which they occur are unique to each memory technology. Reliability issues for FRAMs include retention, fatigue, aging, imprint, reducing-environment, and radiation [14]. The reliability issues of EEPROMs include stuck bit, retention degradation, read time degradation, erase time degradation, program time degradation, disturbs, overerase, erase disturb, program disturb, and read disturb [17].

3.1 Reliability Issues of FRAM

The major non-fabrication-related issues of FRAM reliability include data retention, fatigue, aging, imprint, and radiation. *Data retention*, one of the most important characteristics of non-volatile memories, is defined as the ability of a memory to maintain stored data between the time it is written and the time it is subsequently read. Although data retention is influenced at a fundamental level by design and manufacturing factors, retention failures are accelerated by high temperatures, which cause thermal depolarization of the poled state in the ferroelectric material. The signal loss due to data retention failures recovers after a rewrite and immediate read.

Fatigue occurs in ferroelectric materials with an increased number of switching cycles (read or write cycles) and is characterized by a decrease in switchable polarization. This process is related to the electrode interfacial areas of the memory cells and electric-field assisted migration of oxygen vacancies within ferroelectric materials.

Aging is similar to retention failure in that it is characterized by signal loss over time, but, unlike retention failures, failures due to aging occur during the retention period and do not

recover after a rewrite and immediate read. During the aging process, a gradual stabilization of the domain structure occurs, which causes the ferroelectric material to become less responsive to applied electric fields.

Imprint is a reliability issue specific to ferroelectric material. Accumulation of charge in the ferroelectric cell over time makes a capacitor that has spent a significant amount in one polarity reluctant to switch polarities.

The *radiation tolerance* of ferroelectric memory is limited by the CMOS circuit elements. Prior studies have shown no significant difference between the radiation tolerance of commercial memory devices with and without ferroelectric material.

3.2 Reliability Issues of EEPROM

The reliability issues with EEPROM are very similar with the exception of imprint, which is specific to FRAM. In addition, the process by which fatigue occurs differs, and charge-trapping is an aspect specific to EEPROMs. During programming, the control gate of an EEPROM cell is made positive relative to the source-drain area. The floating gate is capacitively coupled to the control gate, and when sufficient voltage is generated and the tunneling threshold is exceeded, electrons tunnel through the thin "tunnel" oxide window into the floating gate. The negative charge then remains trapped in the floating gate since inadequate voltage exists, normally to allow the electrons to tunnel back out. To erase the memory cell, the process is simply reversed. To read the cell, the control gate and source are brought to predetermined reference voltages and the current through the cells is measured. The transistor of a programmed cell is "on" and the transistor of an erased cell is "off".

Two basic types of failure occur when EEPROM cells are repeatedly written and erased: dielectric failure and charge trapping. Dielectric failures are the source of very low level random failures. They are caused by leakage through minor unscreenable flaws in the tunnel oxide. On contemporary production EEPROMs, dielectric failures are typically too rare to be noticed by standard lot sampling techniques until several hundred thousand write-erase cycles have passed. After this, they create a very low but visible level of random bit failures.

Charge trapping is the effect that creates intrinsic failure in EEPROMs. During write-erase cycling, small amounts of isolated negative and positive charge become trapped in imperfections in the tunnel oxide. Once trapped, the charge is no longer free to tunnel out of the oxide. In practice, electrons are more commonly trapped, and their presence creates a barrier to the tunneling of other electrons through the tunnel oxide. The apparent voltage needed to tunnel in either direction through the oxide increases. This reduces the amount of charge that can be moved in and out of the floating gate. When the accumulation of trapped charge becomes severe enough, it is no longer possible to move enough charge to clearly distinguish a one from a zero. At this point, the memory cells affected must be abandoned.

It is desirable to be able to program EEPROMs as quickly as possible. However, accelerating the programming of EEPROM cells requires the use of higher programming voltages, which accelerate the charge trapping mechanism and generally degrade the endurance of the EEPROM.

It seems intuitive that tunnel oxide might degrade with endurance cycling and that data retention would suffer as a result. But the effect of cycling on the retention characteristics of EEPROM memory is very slight [19]. That it does occur is not due to increasing leakage through normal tunnel oxide, but to the statistical influence of the random failures which are in

fact caused by leakage through rarefied defects. The effect of cycling on the retention characteristics before reaching the intrinsic limit of EEPROM memories is so slight, in fact, that it is usually ignored.

4 FAULT MODELS

The assumption that faults behave according to a particular fault model helps to characterize certain types of errors. Fault models define the types of faults that will be considered and the behavior these faults will have. Also, we can represent the behavior of physical occurrences. Several defects are usually mapped to one fault model.

From the physical point of view, the EEPROM, SRAM, and FRAM memory mechanisms are completely different from one another. However, from the test point of view, EEPROM and FRAM memories are tested with RAM test algorithms using a classical memory functional fault model [22]. In prior work, only Mohammed et al. [23] has presented a unique test algorithm for EEPROMs, but his fault model was specific to flash EEPROMs. Other work has focused on BIST solutions for nonvolatile memories [24]. Ferroelectric memories have a unique fault characteristic referred to as imprint. Imprint is a reliability issue specific to ferroelectric material. Accumulation of charge in the ferroelectric cell over time makes a capacitor that has spent a significant time in one polarity reluctant to switch polarities. This characteristic of FRAMs can be modeled using the stuck-at fault model, which is described below, along with other fault models. Van de Goor [22] also provides a description of the relationship between his reduced functional fault model and the full functional faults they cover. The stuck-at fault covers cell stuck, driver stuck, read/write line stuck, chip-select line stuck, data line stuck, and open in data line faults. Coupling faults cover a short between the data lines and crosstalk between data lines. Address faults cover address line stuck, open in address line, shorts between address lines, open decoder, wrong access, and multiple access faults. Transition faults cover faults that occur when a cell can be set to 0 but not to 1, and vice-versa. Neighborhood pattern-sensitive faults cover pattern-sensitive interactions between cells.

The reduced functional model for conventional memory testing, which is also applicable to EEPROMs and FRAMs, was developed by A.J. Van de Goor [22]. Used for functional testing, the model contains three blocks: the address decoder, the memory cell array, and the read/write logic, as illustrated in Figure 5.

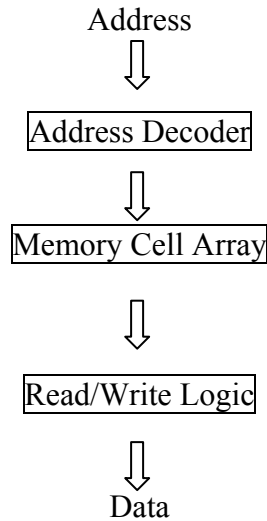


Figure 5 Reduced Functional Model

The reduced functional fault model includes four types of faults: the stuck-at fault, transition fault, coupling fault, and the neighborhood pattern-sensitive fault.

4.1 Stuck-at Fault

The stuck-at fault is defined as follows: the logic value of a faulty cell or line is always 0 (a stuck-at 0 fault) or 1 (a stuck-at 1 fault). A test to detect and locate all stuck-at faults must satisfy the following requirement: from each cell, a 0 and a 1 must be read.

4.2 Transition Fault

A special case of the stuck-at fault is the transition fault, which is defined as follows: a cell or line which fails to undergo a zero-to-one transition when it is written is said contain an up transition fault. Similarly, a down transition fault is the inability for a memory cell to make a one-to-zero transition. A transition fault can be thought of as a stuck-at fault on the set or reset input of an S/R flip-flop. When the set input is stuck-at 0, the fault may be classified as an up transition fault because the S/R flip-flop will fail to make an up transition. If a cell has an up transition fault and is in the 0 state upon power-up, it is effectively a stuck-at 0 fault. When it is in state 1 on power-up, it can undergo a one-to-zero transition. The same argument can be used for down transition faults. However, transition faults cannot be treated as stuck-at faults because other faults, such as coupling faults, may bring the cell back into the opposite state. A test to detect and locate all transition faults should satisfy the following requirement: each cell must undergo an up transition and a down transition, and be read after each transition before undergoing any further transitions.

4.3 Coupling Fault

The types of coupling faults considered are based on the following assumptions: a read operation will not cause a read error, a nontransition write operation will not cause a fault, and a transition write operation may cause a fault. Four types of bridging faults are considered in this fault model: inversion, idempotent, bridging, and state coupling faults. All of these faults fall under the category of 2-coupling faults, which are defined as follows: a write operation which generates an up or a down transition in one cell changes the contents of another cell. Before describing the four faults, some terminology should be understood: when a transition in

cell j can cause a coupling fault in cell i , cell i is said to be coupled to cell j , while j is called the coupling cell.

An inversion coupling fault is defined as follows: an up (or down) transition in one cell inverts the contents of a second cell. A test to detect all inversion coupling faults should satisfy the following requirement: for all cells which are coupled cells, each cell should be read after a series of possible inversion coupling faults may have occurred, with the condition that the number of transitions in the coupling cell is odd (this is so that multiple inversion coupling faults do not mask each other).

An idempotent coupling fault is defined as follows: an up (or down) transition in one cell forces the contents of a second cell to a certain value, 0 or 1. With two possible types of transitions (up transitions and down transitions) and two possible values which the second cell can take on, four possible idempotent coupling faults exist. A test to detect all idempotent coupling faults should satisfy the following requirement: for all cells which are coupled cells, each cell should be read after a series of possible idempotent coupling faults may have occurred in such a way that the sensitized idempotent coupling faults do not mask each other.

A bridging fault, sometimes referred to as a short, consists of a galvanic connection (called a bridge) between two or more cells or lines. It is a bidirectional fault in which the state of a line or cell causes a fault, which makes it different from inversion and idempotent coupling faults in which the fault is caused by a transition. Two types of bridging faults can occur (but not simultaneously): the AND bridging fault and the OR bridging fault. In the AND bridging fault, the logic value of the bridge is the AND of the shorted cells or lines, while in the OR bridging fault the logic value of the bridge is the OR of the shorted cells or lines.

4.4 Neighborhood Pattern-Sensitive Fault

A pattern-sensitive fault is defined as follows: the contents of a cell, or the ability to change the contents, is influenced by the contents of all other cells in the memory. These contents consist of a pattern of 1's and 0's, or changes in these contents.

	d	
D	b	d
	d	

Figure 6 Neighborhood Pattern-sensitive Fault Terminology

Figure 6, which depicts a memory array, illustrates the terminology associated with neighborhood pattern-sensitive faults, where b is the base cell and d is a deleted neighborhood cell, and $b+d$ is defined as the neighborhood. The neighborhood with the base cell excluded is called the deleted neighborhood. The pattern-sensitive fault model allows the deleted neighborhood to take on any position in the memory array. When the deleted neighborhood is allowed to take on only a single position (such that it surrounds the base cell), it is referred to as a neighborhood pattern-sensitive fault (NPSF). Three types of NPSFs are present in Van de Goor's fault model: active, passive, and static. In an active NPSF, the base cell changes its contents due to a change in the deleted neighborhood pattern. This change consists of a transition in one deleted neighborhood cell, while the remaining deleted neighborhood cells and the base cell contain a certain pattern. A test to detect and locate all active NPSFs should satisfy the following requirement: each base cell must be read in state 0 and in state 1 for all possible changes in the deleted neighborhood pattern. In a passive NPSF, the content of the base cell cannot be changed (it cannot make a transition) due to a certain neighborhood pattern. A test to

detect and locate passive NPSFs should satisfy the following requirement: each base cell must be written and read in state 0 and in state 1, for all permutations of the deleted neighborhood pattern. In a static NPSF, the content of a base cell is forced to a certain state due to a certain deleted neighborhood pattern. A test to detect and locate static NPSFs should satisfy the following requirement: each base cell must be read in state 0 and in state 1, for all permutations of the deleted neighborhood pattern.

5 TESTER DESIGN

5.1 Design Choices and Design Flow

A custom memory tester was designed with the goal of creating a low-cost, user-customizable testbench for nonvolatile memory that could perform reliability and endurance tests. A Xilinx XC4010E 10 000-gate, 5-V FPGA was chosen for the 5-V memories, while the Xilinx XC4010XL, a 3.3-V version of the XC4010E, is suitable for use with 3.3-V memories. An XESS XS40 FPGA prototyping board was chosen to implement the tester because, in addition to the FPGA, it contains a parallel port for communication with a PC, an LED for error readout, and an EEPROM socket for PC-independent operation. The Xilinx Foundation ISE tools were used to compile the design. The tester was designed to interface the FPGA with the address and data pins of the memory to perform test-related read and write operations. An error is identified by comparing read data to its known correct value, which is stored in the FPGA. A seven-segment LED is used to display error information.

The design flow used for designing the memory tester using the FPGA is illustrated in Figure 7. The first step, obtaining the memory specifications, was completed by referring to the data sheets for the memory to be tested. The inputs and outputs to be defined include the memory address and data pins, as well as the error output pins to the seven-segment LED display or parallel port, and any switches involved in the tester operation. Design entry was done solely using the Verilog hardware description language. The Verilog code for each tester and memory test is organized into one main module which is the interface to the memory, and two submodules: the memory controller module (which is essentially a large state machine that performs the reads and writes to the memory, implements the test algorithm, and sends data to the LED display module), and the LED display module (which converts BCD and drives the

LED display). The Verilog code for the Ramtron FM24C04 serial FRAM contains an additional module, which divides the XS40 board's clock and uses it to drive the FRAM clock input.

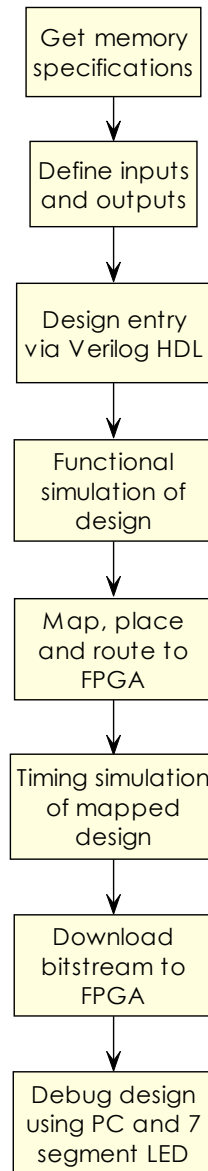


Figure 7 FPGA Design Flow

A functional simulation was then performed using the Xilinx Foundation software in order to verify correct functional operation of the state machine. The design then went through the implementation phase, in which the Xilinx Foundation implementation tools compiled the

netlist into a bitstream used to program the FPGA. The implementation requires mapping the circuit to the specific Xilinx XC4010E FPGA architecture, placing the gates in specific configurable logic blocks (CLBs), and routing the interconnects using the programmable switch matrices (PSMs). A timing simulation was then performed on the mapped design to verify that it meets the timing specifications of the memory. Finally, the bitstream file is downloaded to the FPGA and the debugging process begins. In order to quickly debug the design, extra I/O pins on the FPGA were used to output the current state of the finite state machine (in binary format), as well as other important information related to timing.

Separate test programs were written in Verilog and synthesized using Xilinx Foundation ISE for three types of nonvolatile memories: Ramtron FM24C04 FRAM, Ramtron FM1808 FRAM, and Northrop-Grumman W28C256 Rad-Hard EEPROM. Some relevant device utilization statistics for the Ramtron FM1808 FRAM endurance test code are listed in Table 1. These statistics were obtained from the Xilinx Foundation software after implementation was complete.

Table 1 Device Utilization Statistics

Description	# Used / Total	Percentage Used
External IOBs	51 / 61	83%
Global Buffer IOBs	1 / 8	12%
CLBs	317 / 400	79%

5.2 Memory-Specific Tester Design Issues

The Ramtron FM24C04 is a 4-kb serial FRAM organized as 512 words by 8 bits [25]. The endurance specification of this memory is 10^{10} read or write cycles. This memory uses a two-transistor, two-capacitor structure and is accessed using an industry standard two-wire interface. When accessing the memory, the user addresses 512 locations each with 8 data bits.

These data bits are shifted serially. The 512 addresses are accessed using the two-wire protocol, which includes a slave address (to distinguish other devices) a page address, and a word address. The word address contains 8 bits and the page address contains 1 bit, forming 2 pages with 256 locations per page. The bus protocol is controlled by transition states in the serial data/address (SDA) line with respect to the serial clock (SCL) line. These transitions indicate one of four conditions including start, stop, data bit, and acknowledge. Interconnections between the FPGA and the serial FRAM were simple, with power, ground, and the SDA and SCL lines connected. Timing was more complicated, however, with the requirement that the SDA line change in the middle of a clock pulse to indicate a start or stop condition. This was accomplished with the use of multiple counters to subdivide each clock cycle into smaller parts. The memory was tested at 90% of its maximum rated clock speed of 400 kHz.

The Ramtron FM1808 FRAM is a 256-kb, parallel (byte-wide) FRAM organized as 32,768 words by 8 bits [26]. The endurance specification of this memory is 10^{10} read or write cycles. Like the Ramtron FM24C04, this memory also uses a two-transistor, two-capacitor structure and is accessed through a parallel interface. The 130-ns cycle time is common to both the read and write operations, and the memory has a 70-ns access time. Writes occur immediately at the end of the access with no delay. Unlike an EEPROM, it is not necessary to poll the device for a ready condition since writes occur at bus speed. A precharge operation is required as a part of every memory operation. Interconnections between the FPGA and the parallel FRAM include the power and ground lines, as well as 15 address lines, 8 data lines, and chip, output, and write enable lines. Interconnect capacitances severely limited the speed of operation of the tester on a protoboard, so a PCB board was designed. In spite of using a layout technique to minimize interconnect capacitance, the PCB board also experienced speed limitations. Therefore, the chip

cycle time was increased from 130 ns to 650 ns, which is still within the memory's timing specifications.

The Northrop Grumman W28C256 EEPROM is a 256-kb, parallel (byte-wide), radiation hardened, SONOS-based EEPROM constructed using a mature dual-well CMOS process utilizing N on N⁺ epitaxial silicon and a two-layer interconnect system [7]. The memory is organized as 32 768 words by 8 bits, and a page consists of 64 words. The endurance specification of this memory is 10 000 write cycles with an associated data retention of 10 years and a total dose radiation exposure of between 0 and 50 krad (Si). The programming time for the W28C256 is controlled by an internal counter and an externally supplied clock input. The nominal timing is for a 10 ms programming time with a 2-MHz clock input. All or a portion of the 64-byte page may be loaded prior to writing, but the entire page is always written with the contents of the data latches. Single-byte data modification is not supported. Data polling is used to verify the completion of programming. If a read is performed on any address while the part is still being programmed, the ones complement of the last byte written will be presented at the outputs. After programming has been completed, a read of the last address written will result in the correct data being presented at the outputs. Interconnections between the FPGA and the Rad-Hard EEPROM include the power and ground lines, as well as 15 address lines, 8 data lines, chip, output, and write enable lines, a separate programming supply voltage, and a clock input. This memory was tested using a programming clock speed of 2-MHz, which provides a 10-ms write cycle time.

5.3 Tester Operation

The tester can be configured to perform reliability or endurance tests, and each test can log errors in one of two ways. The error data can either be logged on a PC through the parallel port, or the tester can be used by itself, independent of a PC, by using an EEPROM to load the bit stream file and “scrolling” the error information on a seven-segment LED display. Upon an error in during either test, the tester logs the total number of errors that have occurred, the number of read and write cycles at the point the error occurred, the memory address at which error occurred, the incorrect data value read, and (on the reliability test) it indicates the part of the memory test in which the failure occurred. Two switches are provided for reset and error display functions. The reset switch reset the state machine and the error log, while the error display switch controls the readout of error information. If the parallel port version of the tester is used, a simple program logs the data on the screen and gives the user the option of saving the data to a file. Details of the operation of the memory tester can be found in Appendix A. The memory tester configuration for a parallel FRAM is shown in Figure 8.

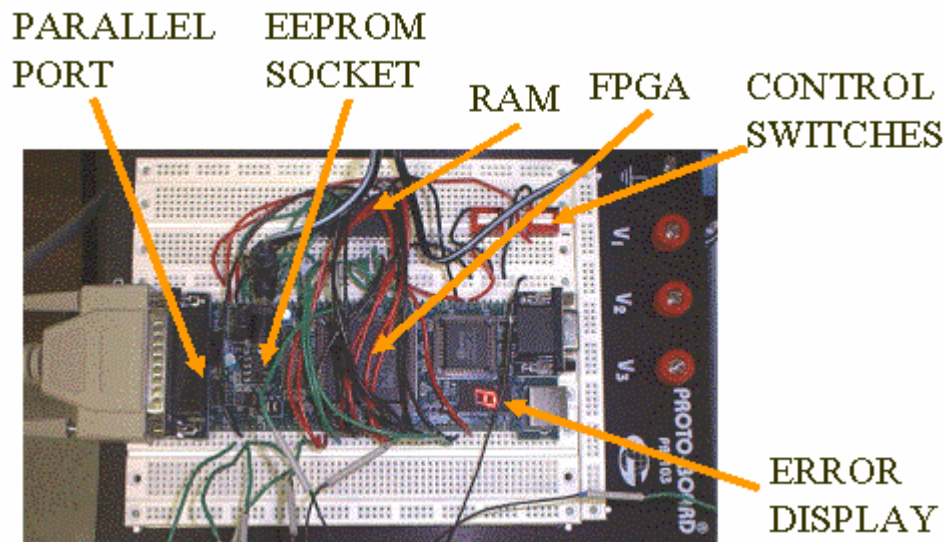


Figure 8 Nonvolatile Memory Tester

This test bench offers several advantages over commercial testers when used for reliability and endurance testing. Endurance testing to a chip's specifications could involve more than 10^{10} read/write cycles, which can take up to 28 days for the Ramtron FM24C04 serial FRAM. Commercially available memory testers with high hourly rates may prove extremely expensive for testing nonvolatile memories with 10^{12} to 10^{15} read/write cycles. In comparison, the FPGA-based testers are inexpensive and more flexible. If several FPGA boards are used, many chips can be tested simultaneously at a fraction of the cost of commercial testers. The highly portable, PC-independent nature of the test bench would also make it suitable for use in radiation or accelerated aging heat testing, assuming appropriate shielding is provided for the tester.

5.4 Tester Design Modification

Modifying the testers to test additional memory types or to perform additional memory tests is relatively easy with a working knowledge of Verilog and finite state machines. The Verilog code for each tester and memory test is organized into one main module which is the interface to the memory, and two submodules: the memory controller module (which is essentially a large state machine that performs the reads and writes to the memory, implements the test algorithm, and sends data to the LED display module), and the LED display module (which converts BCD and drives the LED display). The Verilog code for the Ramtron FM24C04 serial FRAM contains an additional module, which divides the XS40 board's clock and uses it to drive the FRAM clock input.

6 TEST METHODOLOGY

6.1 Reliability Testing

A MATS+ test was chosen to test the reliability of the nonvolatile memories. In order to understand the test procedure, a brief example and explanation of Van de Goor's memory test notation is provided below [22]:

```
UP(W10101010; R; W01010101)
```

- UP = Perform the entire set of operations in parentheses from the first memory address to the last
- W10101010 = Write the data pattern '10101010'
- R = Read back the data
- W01010101 = Write the data pattern 01010101
- (Increment address and loop)

The MATS+ test, which is often used when the memory technology is unknown, is a reliability test that can detect address decoder faults and stuck-at faults. This reliability test was chosen because it met the minimum test criteria while fitting into the relatively small FPGA. Again using Van de Goor's notation, the MATS+ test is described as follows:

1. UP(W01010101)
2. UP(R; W10101010)
3. DOWN(R; W01010101)
4. LOOP BACK TO (2)

The fault coverage of a MATS+ test can be shown informally as follows [22]:

i) For each cell, both data values have been written and verified. This assures that the read and write operations can be applied to each cell and that the cell is not stuck-at.

ii) The march element with the increasing address order verifies that writing into the current cell does not affect a cell with a higher address, because the contents of the latter cell are verified later on by that march element.

iii) The march element with the decreasing address order verifies that writing into the current cell does not affect a cell with a lower address.

iv) From (ii) and (iii), it can be concluded that writing into a particular cell does not affect any other cell. From (i) it can be concluded that the addressed cell can be accessed; hence, the address decoder is correct.

v) The fault coverage of a MATS+ test has been shown by (i) for stuck-at faults and by (iv) for address faults.

6.2 Endurance Testing

In order to test the endurance of the nonvolatile memories, the following basic endurance test was used:

1. (W01010101; R; W10101010; R)
2. LOOP BACK TO (1)

Due to the prohibitive amount of time required to exhaust all addresses in the memories with an endurance test while meeting or exceeding the endurance specification, it was decided that a single address would be used instead. It is important to note that the FRAMs tested

operate with a read and restore mechanism similar to DRAMs, so both read and write operations must be included in the number of endurance cycles counted. This does not apply to EEPROMs.

6.3 MINVDD Testing

While the actual storage material differs in the FRAM and EEPROM memories tested, the address decoder and sense amplifier circuitry are still primarily CMOS-based. Therefore, recently introduced MINVDD test techniques were applied to test for memories weak CMOS circuitry. A weak chip is defined as one that contains flaws, which are defects that do not interfere with correct circuit operation under normal conditions but may cause intermittent or early-life failures. To avoid such early failures, it is necessary to screen out weak chips. Common techniques for detecting flaws in a chip include burn-in, IDDQ testing, high-voltage stress, and low-voltage testing. Prior work has shown that flaws can be accelerated in a low-voltage environment [21]. The *minvdd* of a chip is defined as the minimum supply voltage at which the chip can produce correct logic states at the outputs. In *minvdd* testing, the *minvdd* of a chip is used to decide if a chip is good or weak. Low-voltage testing can be applied at-speed or at a slow speed. At-speed testing can better detect delay flaws, but requires repeated characterizations to account for process variations across different lots. Slow-speed testing requires fewer test runs, but this technique might miss delay flaws that can be detected with an at-speed test. At-speed tests require high transition fault coverage, while slow-speed tests require high stuck-at fault coverage.

7 EXPERIMENTAL RESULTS

Memory testing is ongoing, and the results to date are preliminary, and on a very small sample of chips. Three nonvolatile memories are under test: Ramtron FM24C04 serial FRAM, Ramtron FM1808 Parallel FRAM, and the Northrop-Grumman's 256-kb Rad-Hard EEPROM. Reliability tests have produced no errors in any of the memories. Endurance testing on the Ramtron FM24C04 serial FRAM has exceeded the endurance specification of the chip (10^{10} read/write cycles) by over fourteen times (it has undergone over 14.708×10^{10} read/write cycles) with no errors. Endurance testing on the Ramtron FM1808 parallel FRAM has exceeded the endurance specification of 10^{10} by almost nineteen times, but no errors have surfaced to date. The Northrop-Grumman W28C256 Rad-Hard EEPROM has exceeded its endurance rating by almost 1.5 times with no errors. These results are summarized in Table 2.

Table 2 Preliminary Results

Memory Type	Status	# R/W Cycles (Endurance)	Endurance Spec
Ramtron FM24C04	Testbed Complete	1.47E+11	1.00E+10
Ramtron FM1808	Testbed Complete	1.90E+11	1.00E+10
NG W28C256	Testbed Complete	14 294	10 000

The slow-speed test method was chosen for MINVDD testing for two reasons. First, the limited speed of the parallel FRAM and EEPROM testers due to interconnect capacitance issues made at-speed testing impossible for certain memories. Second, the slow-speed test requires a high stuck-at fault coverage, which is provided by the MATS+ reliability test. The minvdd test was performed by running a MATS+ reliability test on the chip to be tested, starting with a low supply voltage that would produce an error, and increasing the voltage in 0.1-V increments until no errors occurred. The minvdd value was averaged for each memory using samples of 10 Ramtron FM1808 Parallel FRAMs, 10 Ramtron FM24C04 Serial FRAMs, and 3 Northrop Grumman W28C256 EEPROMs. The minvdd for the Ramtron FM24C04

serial FRAM was found to be 3.4 V, while the minvdd for the Ramtron FM1808 Parallel FRAM was found to be 2.1 V. No weak chips were found using these minvdd values.

8 CONCLUSIONS AND FUTURE WORK

The goal of the NASA NEPP-funded project is to determine the reliability and endurance characteristics of certain nonvolatile memories for use in space applications. A low-cost, highly-flexible FPGA-based test bench has been developed for such testing. Preliminary reliability and endurance characterization of a small sample of parallel and serial FRAMs and rad-hard EEPROMs have been completed. However, the reliability and endurance characterization of these memories is in a preliminary state. There is still considerable work left to do before the research is complete:

- Further reliability and endurance testing should be completed using a larger sample of chips from different lots.
- The nonvolatile memories should undergo accelerated aging by heating and high-voltage testing in order to test data retention characteristics. This would also allow testing for imprint in ferroelectric memory by following the test methods described in [9].
- Radiation testing can be performed using this test bed, assuming appropriate shielding is provided for the FPGA.
- It may be useful to add additional reliability test routines in the Verilog program to test for additional faults. Due to the size limitation of the current FPGA, a larger FPGA would be required.
- In the MINVDD testing, the characterization of the minvdd value requires the use of many samples of chips from different lots. More accurate results could be obtained with a larger sample size of chips from different lots.

This test bench offers several advantages over commercial testers when used for reliability and endurance testing. Endurance testing to a chip's specifications could involve more than 10^{10} read/write cycles, which can take up to 28 days for the Ramtron FM24C04 serial FRAM. Commercially available memory testers with high hourly rates may prove extremely expensive for testing nonvolatile memories with 10^{12} to 10^{15} read/write cycles. In comparison, the FPGA-based testers are inexpensive and more flexible. If several FPGA boards are used, many chips can be tested simultaneously at a fraction of the cost compared to the commercial testers. The highly portable, PC-independent nature of the test bench would also make it suitable for use in radiation or accelerated aging heat testing, assuming appropriate shielding is provided for the tester.

REFERENCES

- [1] NASA JPL Technical Staff, "NASA electronic parts and packaging program proposal," National Aeronautics and Space Administration, 2000.
- [2] S. Philpy et al., "Reliability of ferroelectric memory for high-rel and space applications," Celis Semiconductor Corporation, 2001.
- [3] Ramtron Technical Staff, "FRAM technology backgrounder" Ramtron International Corporation, 2001.
- [4] R. Moazzami, C. Hu, and W. Shepherd, "Endurance properties of ferroelectric PZT thin films," University of California, Berkeley, 2001.
- [5] C.Y. Chang and S. M. Sze, Eds., *ULSI Devices*. New York, NY: John Wiley & Sons, Inc., 2000.
- [6] C. Hu, Ed., *Nonvolatile Semiconductor Memories: Technologies, Design, and Applications*. Piscataway, NJ: IEEE Press, 1991.
- [7] Northrop Grumman Corporation, "W28C256 256k rad-hard EEPROM datasheet," Northrop Grumman Corporation, 2001.
- [8] M. White, D. Adams, and J. Bu, "On the go with SONOS," *IEEE Trans. On Circuits and Devices*, pp. 22-30, 2000.
- [9] T. Hadnagy et al., "Retention after fatigue of ferroelectric memories," *Integrated Ferroelectrics*, vol. 37, pp. 215-223, 2001.
- [10] T.D. Hadnagy, S.D. Traynor, D.I. Dalton, "The use and design of experiments to evaluate the reliability of ferroelectric nonvolatile memories," *Integrated Ferroelectrics*, vol. 16 pp. 219-227, 1994.
- [11] Ramtron Corporation, "About FRAM Memory," 2002, <http://www.ramtron.com/aboutfram/aboutfram.htm>.

[12] S.D. Traynor, S. Sun, and D. Hadnagy, "Remanence polarity effects on hydrogen damage of ferroelectric thin film capacitors," *Integrated Ferroelectrics*, vol. 27, pp.195-203, 1999.

[13] Ramtron Corporation, "Ramtron FM24C16 data retention characterization," Ramtron Corporation, 2001.

[14] A. Sharma, A. Teverovsky, "Preliminary evaluation of data retention characteristics for ferroelectric random access memories (FRAMs)," NASA Goddard Space Flight Center, 2001.

[15] D.E. Fisch et al., "Analysis of thin film ferroelectric aging," National Semiconductor Corporation, 2000.

[16] E.M. Philofsky, "FRAM- The ultimate memory," *IEEE Conference on International NonVolatile Memory Technology*, 1996, pp. 99-104.

[17] S. Philpy et al., "Reliability of reprogrammable nonvolatile memories," Celis Semiconductor Corporation, 2001.

[18] P. Canet et al., "EEPROM programming study- time and degradation aspects" *IEEE Transactions on Electronic Devices*, pp. 846-849, 1999.

[19] Fairchild Semiconductor Corporation, "EEPROM endurance prediction," Fairchild Semiconductor Corporation, 2001.

[20] D.A. Adams, D. Mavisz, J.R. Murray, and M.H. White, "SONOS nonvolatile semiconductor memories for space and military applications," *Aerospace Conference, 2001, IEEE Proceedings*, vol. 5, 2001, pp. 2295-2300.

[21] C. Tseng, R. Chen, P. Nigh, E. McCluskey, "MINVDD testing for weak CMOS ICs," *Proc. IEEE*, 2001, pp. 367-378.

[22] A.J. van de Goor, *Testing Semiconductor Memories*. West Sussex, England: John Wiley & Sons, Ltd., 1991.

[23] M.Gh. Mohammed, K.K. Saluja, and A.Yap, "Testing flash memories," *13th International Conference on VLSI Design*, January 2000, pp. 406-411.

[24] S. Winegarden and D. Pannell, "Paragons for memory test," in *Proceedings IEEE International Test Conference*, 1981, pp. 44-48.

[25] Ramtron Corporation, "Ramtron FM24C04 data sheet," Ramtron Corporation, 2001.

[26] Ramtron Corporation, "Ramtron FM1808 data sheet," Ramtron Corporation, 2001.

APPENDIX A.

NVM TESTER USER'S GUIDE

Introduction

Three FPGA-based nonvolatile memory testers have been developed to perform a MATS+ reliability test and an endurance test. The three testers have been developed to test Ramtron FM24C04 Serial FRAM, Ramtron FM1808 Parallel FRAM, and Northrop-Grumman W28C256 Radiation-Hardened EEPROMs. These testers log data to a single-segment LED display, which displays the total number of errors, the address of the most recent error, and erroneous data read, and the part of the test on which the error occurred (on the reliability test).

Tester Operation

1. INITIAL REQUIREMENTS:

Select the memory to be tested (Ramtron FM24C04, Ramtron FM1808, or Northrop-Grumman W28C256) and the memory test to perform (MATS+ reliability test or endurance test). Set the clock speed on the XS40 board according to Table 3, where the divisor is the value to enter in the "GXSETCLK" program, which is part of the XESS Tools software.

Table 3 Clock Speed Settings

Memory Type	Clock Speed (MHz)	Divisor
Ramtron FM24C04	1.2	84
Ramtron FM1808	20	5
NG W28C256	5	20

2. LOADING THE BITSTREAM FILE:

a) If the tester came with preprogrammed EEPROMs for each memory type and test, insert the appropriate EEPROM into the socket of the XESS XS40 FPGA board (consult XESS documentation for details):

- i. Remove jumper J4.
- ii. Insert appropriate EEPROM into EEPROM socket on XS40 board (see XESS documentation for details).

- iii. The FPGA will automatically load the bitstream upon powerup.
- b) If the tester did not come with preprogrammed EEPROMs, you must load the FPGA with the appropriate bitstream via a PC parallel port (consult XESS documentation for details).
 - i. Make sure jumper J4 is installed.
 - ii. Install the XESS XSTOOLS software on a PC with WinNT/98/2k/XP.
 - iii. Connect the parallel port cable between the PC and the XESS XS40 board as described in the XESS documentation.
 - iv. Use the XSLOAD software to download the appropriate bitstream to the FPGA.
 - v. You may remove the parallel port cable, but note that the bitstream will have to be reloaded if power is removed.

3. STARTING THE TEST:

- a) Make sure the reset switch is in the 'ON' or '1' position.
- b) Make sure the error switch is in the 'OFF' or '0' position.
- c) Flip the reset switch to the 'OFF' position to begin testing.

4. READING THE LED DISPLAY (ERROR DISPLAY):

During testing, under normal operation while no errors have occurred, the LED display will display a dash to indicate testing is in progress with no errors. If an error occurs at any point during testing, the display will change to a 'P' while testing continues. When the error switch is flipped to the 'ON' position, the single-segment LED display "scrolls" error information across the display one digit at a time, with an underscore displayed to separate each displayed number or character. A character or symbol, which will be referred to as an "indicator," is first displayed for a relatively long time prior to each category of data being displayed. This is done to indicate the category of information that is about to be displayed. The order of the displayed information, along with the symbol displayed to indicate what is being displayed next, is described below:

- a) Indicator = E: Total number of errors so far (displays up to 4095, then stays at 4095); this number is displayed in hexadecimal to save display time.

b) Indicator = A: The memory address of the most recent error; this number is displayed in binary.

c) Indicator = C: The total number of read/write cycles that had taken place when the most recent error occurred, or the total number of read/write cycles that have occurred so far (if no errors have occurred). This number is displayed in hexadecimal to save display time.

d) Indicator = V: The value of the last erroneous data read; this number is displayed in binary.

e) Indicator = (S without top and bottom horizontal lines): The state upon which the error occurred; this number is only relevant when performing a MATS+ test. A '1' indicates the error occurred during the UP(R; W10101010) portion of the test. A '2' indicates an error occurred during the DOWN(R; W01010101) portion of the test.

In order to differentiate a hexadecimal 'B' from the number '8' on the LED display, the hexadecimal 'B' is displayed without the top horizontal line.

Tester Modification

Modifying the testers to test additional memory types or perform additional memory tests is relatively easy with a working knowledge of Verilog and finite state machines. The Verilog code for each tester and memory test is organized into one main module which is the interface to the memory, and two submodules: the memory controller module (which is essentially a large state machine that performs the reads and writes to the memory and implements the test algorithm), the BCD to seven-segment display module (which converts BCD and drives the LED display). The Verilog code for the Ramtron FM24C04 serial FRAM contains an additional module which divides the XS40 board's clock and uses it to clock the FRAM.

APPENDIX B. CIRCUIT SCHEMATICS FOR TESTER

Circuit schematics for the three memory testers are presented in Figures 9, 10, and 11.

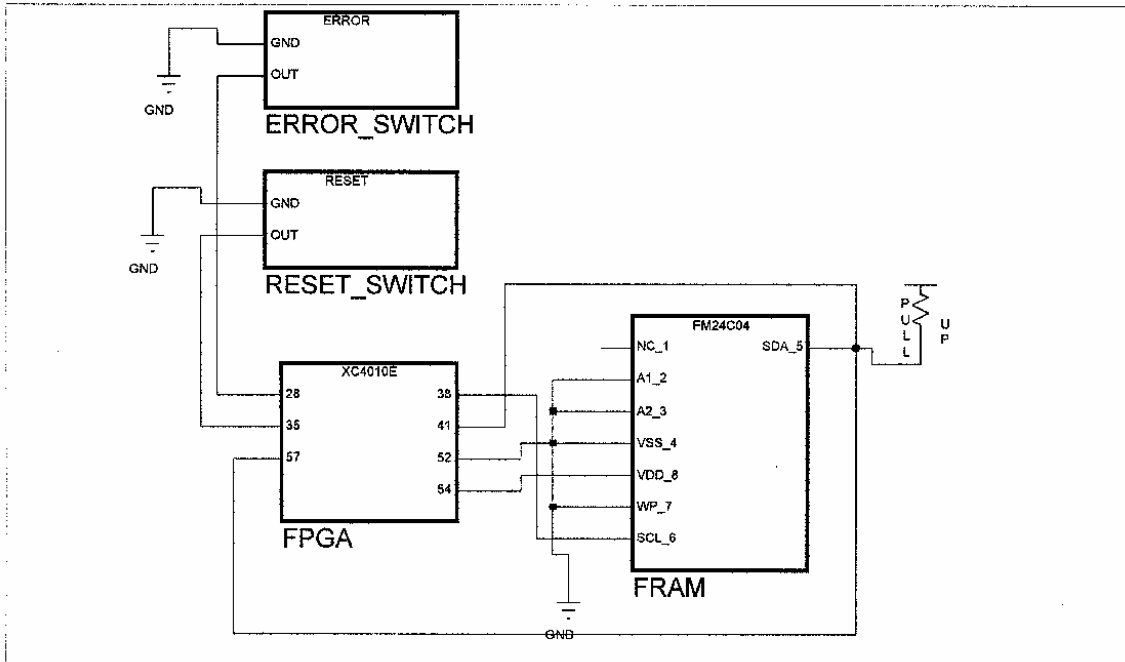


Figure 9 Ramtron FM24C04 Memory Tester Circuit Schematic

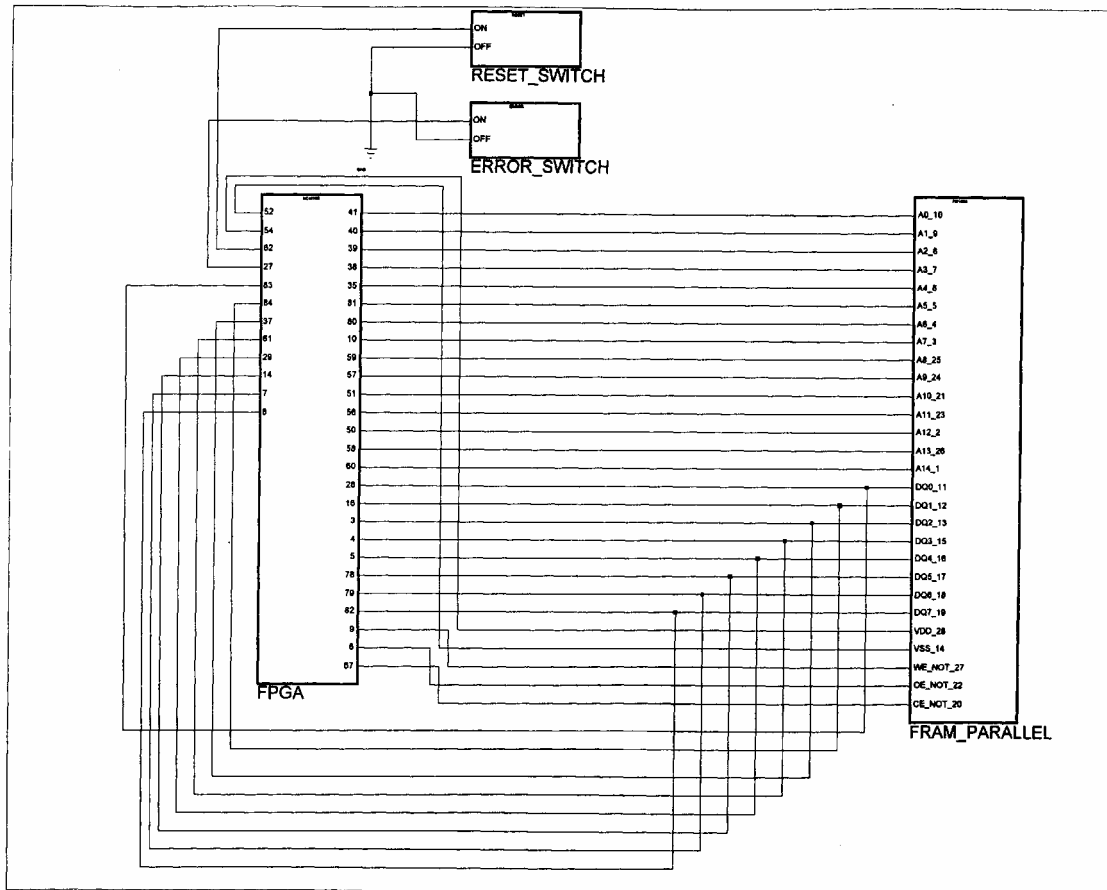


Figure 10 Ramtron FM1808 Memory Tester Circuit Schematic

APPENDIX C.
VERILOG CODE SAMPLE
(MATS+ reliability testing)

```
// MEMORY TESTER VERILOG CODE
// TO TEST: RAMTRON FM1808 PARALLEL FRAM
// WITH TEST: MATS+ RELIABILITY

//AUTHOR: VIKRAM RAO
//UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
//MASTERS STUDENT IN ELECTRICAL AND COMPUTER ENGINEERING

//THIS VERILOG CODE WAS WRITTEN TO PERFORM A MATS+ RELIABILITY TEST ON
RAMTRON
//FM1808 PARALLEL FRAM. PLEASE SEE THESIS "AN FPGA-BASED TESTBENCH FOR
//RELIABILITY AND ENDURANCE CHARACTERIZATION OF NON-VOLATILE MEMORY" FOR
DETAILS.

//BEGIN (TOP MODULE)
module top (clkIn, reset, ucdisable, ramdisable, data_out, RDIN, errorout,
ADDROUT, DATAOUT, WE, OE, CE, sczero, scone, sctwo, sctthree, scfour,
scfive); // scfive, scfour, sctthree, sctwo, scone); BEGIN (TOP)
parameter n = 17;
input clkIn;
input reset;
input [7:0] RDIN;
input errorout;
//OUTPUTS to disable ram, microcontroller
output ucdisable;
output ramdisable;
//output zeropin;
output scfive;
output scfour;
output sctthree;
output sctwo;
output scone;
output sczero;
output [6:0] data_out;
output [14:0] ADDRROUT;
output [7:0] DATAOUT;
output WE;
output OE;
output CE;
wire [2:0] startcount;
wire [4:0] ledin;
//RAM OUTPUTS
wire [14:0] ADDRROUT;
wire [7:0] DATAOUT;
wire WE;
wire OE;
wire CE;

//ASSIGNING VALS FOR UCDISABLE, RAMDISABLE
assign ucdisable = 1;
assign ramdisable = 1;
```

```

//MODULE #1: MEMORY CONTROLLER MODULE
memcontrol mc(.reset(reset), .clk5(clkin), .startcount(startcount),
.RDIN(RDIN), .ledin(ledin), .errorout(errorout), .ADDRROUT(ADDRROUT),
.DATAOUT(DATAOUT), .WE(WE), .OE(OE), .CE(CE), .sczero(sczero),
.scone(scone), .sctwo(sctwo), .scthree(scthree), .scfour(scfour),
.scfive(scfive)); //.scone(scone), .sctwo(sctwo), .scthree(scthree),
.scfour(scfour), .scfive(scfive) .slowclk(slowclk));

//MODULE #2: 17-BIT COUNTER
clkfbcounter5 clkcounter5(.clk5(clkin), .reset(reset),
.startcount(startcount));

//MODULE #3: BCD27SEG (FOR 7-SEGMENT LED DISPLAY)
bcd27seg seg(.AIN(ledin[0]),
                .BIN(ledin[1]),
                .CIN(ledin[2]),
                .DIN(ledin[3]),
                .EIN(ledin[4]),
                .a(data_out[0]),
                .b(data_out[1]),
                .c(data_out[2]),
                .d(data_out[3]),
                .e(data_out[4]),
                .f(data_out[5]),
                .g(data_out[6]));

endmodule //END (TOP)

//MODULE #3: BCD27SEG
module bcd27seg (AIN, BIN, CIN, DIN, EIN, a, b, c, d, e, f, g); // BEGIN 7
SEG DISPLAY
parameter BLANK = 7'b0000000;

//CORRECT DECODER- DISPLAYS 0-9
parameter ZERO = 7'b1110111;
parameter ONE = 7'b0010010;
parameter TWO = 7'b1011101;
parameter THREE = 7'b1011011;
parameter FOUR = 7'b0111010;
parameter FIVE = 7'b1101011;
parameter SIX = 7'b1101111;
parameter SEVEN = 7'b1010010;
parameter EIGHT = 7'b1111111;
parameter NINE = 7'b1111011;
//parameter A already defined
parameter B = 7'b0111111;
//parameter C already defined
parameter D = 7'b1110111;
parameter E = 7'b1101101;
parameter F = 7'b1101100;
//ADDITIONAL OUTPUTS FOR 10-15
parameter A = 7'b1111110; //A = ADDRESS
parameter C = 7'b1100101; //C = CYCLE #
parameter V = 7'b0110111; //V = INCORRECT DATA VALUE

```

```

parameter BETWEEN = 7'b00001111;
parameter END = 7'b1101101;
parameter OK = 7'b0001000;
parameter S = 7'b0101010;
parameter P = 7'b1111100;

input AIN, BIN, CIN, DIN, EIN;
output a, b, c, d, e, f, g;

// COMMENTS ABOUT THE 7-SEGMENT LED ON THE XS BOARD (positive logic)
// S6
// S5 S4
// S3
// S2 S1
// S0
// declarations
reg [6:0] data_out;

// assignments
assign {a, b, c, d, e, f, g} = data_out;

always @ (AIN or BIN or CIN or DIN or EIN)
case ({EIN, DIN, CIN, BIN, AIN})
0: data_out = ZERO;
1: data_out = ONE;
2: data_out = TWO;
3: data_out = THREE;
4: data_out = FOUR;
5: data_out = FIVE;
6: data_out = SIX;
7: data_out = SEVEN;
8: data_out = EIGHT;
9: data_out = NINE;
//ADDITIONAL OUTPUTS FOR SINGLE LED
10: data_out = A;
//11: data_out = C;
11: data_out = B;
//12: data_out = V;
12: data_out = C;
//13: data_out = BETWEEN;
13: data_out = D;
//14: data_out = END;
14: data_out = E;
//15: data_out = OK;
15: data_out = F;
//NEW DEFS
//16: data_out = S;
16: data_out = V;
17: data_out = BETWEEN;
18: data_out = END;
19: data_out = OK;
20: data_out = S;
21: data_out = P;
default: data_out = BLANK;
endcase
endmodule //END (7 SEG DISPLAY)

```

```

//MODULE #2:
CLKFBCOUNTER5=====
// 2. COUNTER: COUNT<3:0> WILL BE USED AS A TIMER TO DETERMINE WHEN TO
PERFORM OPERATIONS
//          IN THE MEMORY CONTROLLER MODULE
module clkfbcounter5 (clk5, reset, startcount); // slowclk); // sctthree,
sctwo, scone);
parameter n = 17;
input clk5, reset;
output [2:0] startcount;
reg [2:0] count;
reg [2:0] startcount;

always @(posedge clk5)
begin
if (reset) begin
startcount <= 0;
count <= 0;
end
else begin
count <= count + 1;
startcount <= startcount + 1; //STARTCOUNT KEEPS TRACK OF STARTING POINT
FOR READS/WRITES
if (startcount > 2) begin
startcount <= 0;
end
end
end //always
endmodule

```

```

//MODULE #1:
MEMCONTROL=====
//MEMORY CONTROLLER MODULE
module memcontrol (ADDRROUT, DATAOUT, WE, OE, CE, reset, clkin, startcount,
RDIN, ledin, errorout, sczero, scone, sctwo, sctthree, scfour, scfive); //
slowclk, scone, sctwo, sctthree, scfour, scfive, );
parameter n = 17;
input reset;
input clkin;
input [2:0] startcount;
input [7:0] RDIN;
input errorout;
//output zeropin;
output sczero;
output scone;
output sctwo;
output sctthree;
output scfour;
output scfive;

```

```

output [4:0] ledin;

//RAM OUTPUTS
output CE;
output WE;
output OE;
output [14:0] ADDR;
wire [14:0] ADDR;
output [7:0] DATA;
wire [7:0] DATA;

    reg CE; //NOT chip enable
    reg WE; //NOT write enable
    reg OE; //NOT output enable
    reg ADDR; //ADDRESS ENABLE
    reg DATA; //DATA ENABLE

    reg [14:0] ADDR; // 8 BITS OF 9-BIT ADDRESS
    reg [7:0] DATA; //8 BITS OF DATA TO WRITE TO RAM
    reg [7:0] CORRECTONE; // VALUE TO COMPARE WITH UPON READ
    reg [7:0] CORRECTTWO;
    reg [24:0] ERRORADDR; // STORES ADDRESS IF ERROR OCCURS
    reg [7:0] DATAREAD;
    reg ERROR;
    reg [1:0] statecount; //state 0 = wo, 1 = (r, w1), 2 = (r,w0)
    reg [43:0] cycle;
    reg [4:0] ledin; //final output for leds
    reg [3:0] ledcount; //state var for leds
    reg [23:0] ledtimer; //timer for led delay
    reg [3:0] binstate; //expanded state var for hex conv
    reg [3:0] cycletemp; //temp var for hex output of cycle #
    reg [3:0] cyclecount; //var for hex conversion
    reg [11:0] errorcount; //counts total # of errors
    reg anyerror; // 1 if any errors have occurred
    reg [43:0] cyclekeep; // counts # of cycles (either total if no error, or
of most recent error if error)
    reg sczero, scone, sctwo, sctthree, scfour, scfive;

integer short_timer;
parameter S0=0, S1=1, S2=2, S3=3, S4=4, S5=5, S6=6, S7=7, S8=8, S9=9,
S10=10, S11=11, S12=12, S13=13, S14=14, S15=15, S16=16, S17=17, S18=18,
S19=19, S20=20, S21=21, S22=22, S23=23, S24=24, S25=25, S26=26, S27=27,
S28=28, S29=29, S30=30, S31=31, S32=32, S33=33, S34=34, S35=35, S36=36,
S37=37, S38=38, S39=39;
reg [5:0] state;
//p is used to increment data bit read out
integer p;

//ASSIGNMENT STATEMENT:
//THIS ELIMINATES THE NEED FOR AN "ENABLE" REG
//IF REGVAL = 1, THEN SDA = 1'BZ
//ELSE SDA = 1'B0
//HOW TO USE THIS:
//- TO SET SDA HIGH *OR* HI-Z, SET REGVAL <= 1; (REPLACE ENABLE = 0 WITH
REGVAL = 1)
//- TO SET SDA LOW, SET REGVAL <= 0 (SAME)

```

```

//- ELIMINATE ENABLE
//assign SDA = (regval ? 1'bz : 1'b0);
//EXAMPLE OF HOW TO USE:
//IF ADDRENABLE = 1, ADDRROUT = ADDR
//IF ADDRENABLE = 0, ADDRROUT = 15'bz

assign ADDRROUT = (ADDRENABLE ? ADDR : 15'bz);
assign DATAOUT = (DATAENABLE ? DATA : 8'bz);

//
//-----
// BIG FINITE STATE MACHINE BEGINS HERE
//-----
always @(posedge clk)
begin

//RESET BEGIN
if (reset == 1) begin
//DATA FOR UP(WO)
    DATA <= 8'b01010101;
    ADDR <= 15'b0000000000000000;
//DEFINE CORRECT VALUE FOR COMPARISON
    CORRECTONE <= 8'b01010101;
    CORRECTTWO <= 8'b10101010;
    statecount <= 0;
    errorcount <= 0;
    cycle <= 1;
    ERROR <= 0;
    anyerror <= 0;
    ledin <= 0;
    cyclekeep <= 0;
    state <= S0;
end

else begin
//OUTPUTTING STATES HERE FOR DEBUGGING PURPOSES, OTHERWISE USE AS EXTRA
PINS!
scfive <= state[5];
scfour <= state[4];
scthree <= state[3];
sctwo <= state[2];
scone <= state[1];
sczero <= state[0];

    case(state)
//#####
//S0: INITIALIZATION FOR (WO)
//#####
        S0: begin
//DATA FOR UP(WO)
            DATA <= 8'b01010101;
            ADDR <= 15'b0000000000000000;
//DEFINE CORRECT VALUE FOR COMPARISON
            CORRECTONE <= 8'b01010101;
            CORRECTTWO <= 8'b10101010;
            statecount <= 0;
            errorcount <= 0;

```

```

        cycle <= 1;
        anyerror <= 0;
        ERROR <= 0;
        state <= S9;
    end //END S0

//#####
//S1: READ
//#####
    S1: begin
    if ((short_timer >= 0) && (short_timer < 4)) begin
        CE <= 1;
        WE <= 1;
        OE <= 0;
        ADDRENABLE <= 0; //ADDRROUT = HI-Z
        DATAENABLE <= 0; //DATAOUT = HI-Z
    end
    else if (short_timer == 4) begin
        ADDRENABLE <= 1; //ADDRROUT = ADDR
    end
    else if (short_timer == 7) begin
        CE <= 0;
    end
    else if (short_timer == 8) begin
        OE <= 0;
    end
    else if (short_timer == 15) begin
        DATAREAD <= RDIN; //read data from RDIN pins and store in DATAREAD reg
    end
    else if (short_timer == 16) begin
        CE <= 1;
        OE <= 1;
    end
    else if (short_timer == 18) begin
        DATAENABLE <= 0; //DATAOUT = HI-Z
        ADDRENABLE <= 0; //ADDRROUT = HI-Z
    end

    if (short_timer < 19) begin
        short_timer <= short_timer + 1;
    end
    else begin
        state <= S2;
    end
end // end S1

//#####
//S2: CHECK FOR ERROR
//#####
//if (short_timer >= 172) begin
    S2: begin // <=====
NEED TO GO THROUGH S4 EVERY TIME TO CHECK SWITCH STATUS!
    if (statecount == 1) begin
        if (DATAREAD != CORRECTONE) begin
            if (errorcount < 4095) begin

```

```

        errorcount <= errorcount + 1;
    end
    else begin
        errorcount <= 4095;
    end
    ERROR <= 1; //INDICATES IF AN ERROR OCCURRED AT THE CURRENT ADDRESS
        anyerror <= 1; //INDICATES IF ANY ERRORS HAVE OCCURRED
    state <= S4; //OUTPUT TO LED
end
else begin
    ERROR <= 0;
    state <= S4; //DONE WITH READ, NOW DO WRITE
end
end
end

else if (statecount == 2) begin
    if (DATAREAD != CORRECTTWO) begin
        if (errorcount < 4095) begin
            errorcount <= errorcount + 1;
        end
        else begin
            errorcount <= 4095;
        end
        ERROR <= 1;
        anyerror <= 1;
        state <= S4; //SEND TO PARALLEL PORT
    end
    else begin
        ERROR <= 0;
        state <= S4; //DONE WITH READ, NOW DO WRITE
    end
end
end //end main begin

```

```

#####
//S3: INCREMENT ADDR, PAGESEL, AND CYCLE #
#####
    S3: begin
if (statecount == 0) begin //(W0)
    if (ADDR == 32767) begin
//DONE WITH R, W1; NOW DO R, W0
        state <= S16; //IF DONE WITH (W0), INIT FOR (R,W1)
    end
    else begin
        ADDR <= ADDR + 1; //ADDR INCREMENTED IF IT HASN'T REACHED 255
        state <= S9; //RESET FOR ANOTHER WRITE
    end // end else
end //end (if statecount)

```

```

else if (statecount == 1) begin // (R, W1)
    if (ADDR == 32767) begin
#####
//THIS IS WHERE THE CYCLE # IS INCREMENTED!
#####

```

```

    cycle <= cycle + 1;
//DONE WITH R, W1; NOW DO R, W0
    state <= S11; //IF DONE WITH (R,W1), INIT FOR (R,W0)
    end
    else begin
        ADDR <= ADDR + 1; //ADDR INCREMENTED IF IT HASN'T REACHED 255
        state <= S10; //RESET FOR ANOTHER READ
    end // end else
end //end (if statecount)

else if (statecount == 2) begin //(R, W0)
    if (ADDR == 0) begin //IF DONE WITH DOWN(R,W0)
//#####
//THIS IS WHERE THE CYCLE # IS INCREMENTED!
//#####
        cycle <= cycle + 1;
        state <= S16; // IF WE ARE LESS THAN X # CYCLES, GO ON TO NEXT
STATE=UP(R, W1)
    end
    else begin
        ADDR <= ADDR - 1;
        state <= S10; //RESET FOR ANOTHER READ
    end //end else begin
end // end else if (statecount)

//else begin
// state <= S8;
//end
end //END S3

//#####
//S4: SETUP FOR LED OUTPUT
//#####
    S4: begin
//NEW CODE $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
    if (anyerror == 1) begin
        ledin <= 21; //OUTPUT 'P' ON LEDS
        if (ERROR == 1) begin //IF ANY ERROR HAS OCCURRED AND THE CURRENT
ADDRESS IS IN ERROR
            cyclekeep <= cycle; //THEN SAVE THE CURRENT ADDR IN CYCLEKEEP
        end
        else if (ERROR == 0) begin //IF ANY ERROR HAS OCCURRED AND THERE IS NO
ERROR AT THE CURRENT ADDRESS THEN KEEP THE OLD VALUE OF CYCLEKEEP
            cyclekeep <= cyclekeep;
        end
    end
    else begin
        cyclekeep <= cycle; //IF THERE HAVE NOT BEEN ANY ERRORS SO FAR, SAVE THE
TOTAL NUMBER OF CYCLES SO FAR INTO CYCLEKEEP
        ledin <= 19; //OUTPUT '-' ON LEDS
    end

    if (ERROR == 1) begin
        if (statecount == 1) begin

```

```

        ERRORADDR <= {ADDR, DATAREAD, statecount};
    end
    if (statecount == 2) begin
        ERRORADDR <= {ADDR, DATAREAD, statecount};
    end
//SET BINADDER, BINSTATE REGS HERE
//APPEND 2 0'S TO STATE TO MAKE IT 4 BITS FOR HEX CONV
    binstate <= {2'b0, statecount};
end

    cyclecount <= 1; //ALWAYS RESET CYCLECOUNT TO 1 SO THAT IT STARTS AT THE
RIGHT POINT IF SWITCH IS FLIPPED!

//#####CHECK FOR ERROR SWITCH HERE!
    if (errorout == 1) begin
        state <= S15; //ERROR SWITCH IS ON, SO DO LED ERROR OUTPUT
    end
    else begin
        state <= S9; //ERROR SWITCH IS OFF, SO INFO HAS BEEN LOGGED; CONTINUE
WITH WRITE
    end
end

//#####
// S15: LED OUTPUT = 'E'
//#####3
    S15: begin
        ledin <= 18;
        ledtimer <= 0;
        ledcount <= 0;
//4085 = 1111 1111 1111 (bin)
        if (errorcount > 4000) errorcount <= 4095;
        state <= S5;
    end

//#####
//S5: LED OUTPUT DELAY
//#####
    S5: begin
        if (ledtimer <= 5000000) begin
            ledtimer <= ledtimer + 1;
            state <= S5;
        end
        else if ((ledtimer > 5000000) && (ledtimer < 7500000)) begin
            ledtimer <= ledtimer + 1;
            ledin <= 17;
            state <= S5;
        end
        else begin
            state <= S68;
        end
    end

//#####

```

```

//S68: LED OUTPUT: ERROR #
//#####
S68: begin
  if (cyclecount == 1) begin
    cycletemp <= errorcount[11:8];
  end
  else if (cyclecount == 2) begin
    cycletemp <= errorcount[7:4];
  end
  else if (cyclecount == 3) begin
    cycletemp <= errorcount[3:0];
  end
  ledcount <= 0;
  ledtimer <= 0;
  state <= S69;
end

//#####
//S69: LED OUTPUT: error #
//#####
S69: begin
  ledin <= cycletemp;
  cyclecount <= cyclecount + 1;
  state <= S54;
end

//#####
//S51: LED OUT = 'A'
//#####
S51: begin
  ledin <= 10;
  ledtimer <= 0;
  ledcount <= 1;
  state <= S52;
end

//#####
//S52: LED OUTPUT DELAY
//#####
S52: begin
  if (ledtimer <= 5000000) begin
    ledtimer <= ledtimer + 1;
    state <= S52;
  end
  else if ((ledtimer > 5000000) && (ledtimer < 7500000)) begin
    ledtimer <= ledtimer + 1;
    ledin <= 17;
    state <= S52;
  end
  else begin
    state <= S53;
  end
end

```

```

#####
//S53: LED OUTPUT: INIT INDEX FOR 9 BIT ADDR OUTPUT
#####
S53: begin
    ledtimer <= 0;
    p <= 24;
    ledcount <= 1;
    state <= S50;
end

#####
//S50: MAIN OUTPUT FOR ALL 71 BITS OF ERRORDATA
#####3
S50: begin
    ledin <= ERRORADDR[p];
    state <= S54;
end

#####
//S54: MAIN OUTPUT FOR ALL 71 BITS OF ERRORDATA + BETWEEN CHAR
#####
S54: begin
// if (errorout == 1) begin //CHECK IF SWITCH IS STILL ON. IF SO, DO ERROR
OUTPUT
    if (ledtimer <= 1500000) begin
        ledtimer <= ledtimer + 1;
        state <= S54;
    end

    else if ((ledtimer > 1500000) && (ledtimer <= 2500000)) begin
        ledin <= 17; //LED = 'BETWEEN'
        ledtimer <= ledtimer + 1;
        state <= S54;
    end

    else if (ledtimer > 2500000) begin
// short_timer <= 0; //reset short_timer

        if (ledcount == 0) begin //OUTPUT ERROR #
            if (cyclecount > 3) begin
                ledtimer <= 0;
                cyclecount <= 1;
                state <= S51;
            end
            else begin
                state <= S68;
            end
        end

        else if (ledcount == 1) begin //OUTPUT ADDR
            state <= S56;
        end

        else if (ledcount == 2) begin //OUTPUT CYCLE #

```

```

        if (cyclecount > 11) begin
            ledtimer <= 0;
            cyclecount <= 1;
            state <= S61;
        end
        else begin
            state <= S59;
        end
    end
end
else if (ledcount == 3) begin          //OUTPUT INCORRECT DATA READ
    state <= S64;
end
else if (ledcount == 4) begin          //FINAL LED OUTPUT STATE
    if (errorout == 1) begin //CHECK IF SWITCH IS STILL ON
        ledtimer <= 0;
        cyclecount <= 1;
//        errorcount <= 0;
        state <= S15; //DONE WITH 1 RUN OF LED OUTPUT, ERROR SWITCH ON SO
LOOP BACK AND DO LED OUTPUT AGAIN
    end
    else begin
        ledtimer <= 0;
        state <= S9; //DONE WITH 1 RUN OF LED OUTPUT, ERROR SWITCH OFF SO DO
WRITE
    end
end
// else if (ledcount == 5) begin //OUTPUT ERROR #
//
// end
// else if (ledcount == 6) begin //OUTPUT ERROR #
//
// end
// end
// end // if (errorout == 1)
// else begin //IF ERROR SWITCH IS OFF, DO WRITE
//     state <= S9;
// end
end
end

#####
//S56: INCREMENT 9 BIT ADDR READOUT
#####
S56: begin
    if (p > 10) begin
        p <= p - 1;
        ledtimer <= 0;
        state <= S50; //AFTER INCREMENT/DECREMENT, GO BACK TO MAIN OUTPUT
    end
    else begin
        ledtimer <= 0;
        state <= S57; // DONE WITH 9 BIT OUTPUT, OUTPUT NEXT MARKER CHAR = 'C'
    end
end
end

```

```

#####
//S57: LED OUT = 'C'
#####
S57: begin
    ledin <= 12;
    ledtimer <= 0;
    p <= 43; //init for hex output
    cyclecount <= 1;
    state <= S58;
end

#####
//S58: LED OUTPUT DELAY
#####
S58: begin
    if (ledtimer <= 5000000) begin
        ledtimer <= ledtimer + 1;
        state <= S58;
    end
    else if ((ledtimer > 5000000) && (ledtimer < 7500000)) begin
        ledtimer <= ledtimer + 1;
        ledin <= 17;
        state <= S58;
    end
    else if (ledtimer >= 7500000) begin
        state <= S59;
    end
end

#####
//S59: LED OUTPUT: INIT FOR CYCLE READOUT
#####
S59: begin
    if (cyclecount == 1) begin
        cycletemp <= cyclekeep[43:40];
    end
    else if (cyclecount == 2) begin
        cycletemp <= cyclekeep[39:36];
    end
    else if (cyclecount == 3) begin
        cycletemp <= cyclekeep[35:32];
    end
    else if (cyclecount == 4) begin
        cycletemp <= cyclekeep[31:28];
    end
    else if (cyclecount == 5) begin
        cycletemp <= cyclekeep[27:24];
    end
    else if (cyclecount == 6) begin
        cycletemp <= cyclekeep[23:20];
    end
    else if (cyclecount == 7) begin
        cycletemp <= cyclekeep[19:16];
    end
end

```

```

else if (cyclecount == 8) begin
    cycletemp <= cyclekeep[15:12];
end
else if (cyclecount == 9) begin
    cycletemp <= cyclekeep[11:8];
end
else if (cyclecount == 10) begin
    cycletemp <= cyclekeep[7:4];
end
else if (cyclecount == 11) begin
    cycletemp <= cyclekeep[3:0];
end
ledcount <= 2;
ledtimer <= 0;
state <= S60; //just delay (no ledin = erroraddr(p)
// 12: begin
//     ledtimer <= 0;
//     state <= S61;
//     end
end

#####
//S60: LED OUTPUT: INCREMENT FOR 44 BIT CYCLE OUTPUT
#####
S60: begin
    ledin <= cycletemp;
    cyclecount <= cyclecount + 1;
    state <= S54;
end

#####
//S61: LED OUT = 'V'
#####
S61: begin
    ledin <= 16;
    ledtimer <= 0;
    state <= S62;
end

#####
//S62: LED OUTPUT DELAY
#####
S62: begin
    if (ledtimer <= 5000000) begin
        ledtimer <= ledtimer + 1;
        state <= S62;
    end
    else if ((ledtimer > 5000000) && (ledtimer < 7500000)) begin
        ledtimer <= ledtimer + 1;
        ledin <= 17;
        state <= S62;
    end
    else begin
        state <= S63;
    end
end

```

```

        end
    end

//#####
//S63: LED OUTPUT: INIT FOR INCORRECT DATA READOUT
//#####
    S63: begin
        p <= 9;
        ledcount <= 3;
        ledtimer <= 0;
        state <= S50;
    end

//#####
//S64: LED OUTPUT: INCREMENT FOR 8 BIT INCORRECT DATA OUTPUT
//#####
    S64: begin
        if (p>2) begin
            p <= p - 1;
            ledtimer <= 0;
            state <= S50; //AFTER INCREMENT/DECREMENT, GO BACK TO MAIN OUTPUT
        end
        else begin
            ledtimer <= 0;
            state <= S65; // DONE WITH 9 BIT OUTPUT, OUTPUT NEXT MARKER CHAR =
'mod s'
        end
    end

//#####
//S65: LED OUT = 'S'
//#####
    S65: begin
        ledin <= 20;
        ledtimer <= 0;
        state <= S66;
    end

//#####
//S66: LED OUTPUT DELAY
//#####
    S66: begin
        if (ledtimer <= 5000000) begin
            ledtimer <= ledtimer + 1;
            state <= S66;
        end
        else if ((ledtimer > 5000000) && (ledtimer < 7500000)) begin
            ledtimer <= ledtimer + 1;
            ledin <= 17;
            state <= S66;
        end
        else begin
            state <= S67;
        end
    end

```

```

        end
    end

//#####
//S67: LED OUTPUT: INIT FOR STATE READOUT
//#####
    S67: begin
//    p <= 1;
        ledcount <= 4;
        ledtimer <= 0;
        ledin <= binstate;
        state <= S54;
    end

//#####
//S7: WRITE
//#####
    S7: begin
if ((short_timer >= 0) && (short_timer < 4)) begin
    CE <= 1;
    WE <= 1;
    OE <= 0;
    ADDRENABLE <= 0; //ADDRROUT = HI-Z
    DATAENABLE <= 0; //DATAOUT = HI-Z
end
else if (short_timer == 4) begin
    WE <= 0;
    ADDRENABLE <= 1; //OUTPUT ADDR (ADDRROUT = ADDR)
end
else if (short_timer == 7) begin
    CE <= 0;
end
else if (short_timer == 8) begin
    DATAENABLE <= 1; //OUTPUT DATA TO WRITE
end
else if (short_timer == 11) begin
    CE <= 1;
end
else if (short_timer == 12) begin
    WE <= 1;
end
else if (short_timer == 13) begin
    DATAENABLE <= 0; //DATAOUT = HI-Z
    ADDRENABLE <= 0; //ADDRROUT = HI-Z
end

if (short_timer < 14) begin
short_timer <= short_timer + 1;
end
else begin
state <= S3; //INCREMENT
end

end //end S7

```

```

//#####
//S8: IDLE STATE AFTER FINISHED
//#####
    S8: begin
        state <= S8;
    end

//#####
//S9: SHORT_TIMER RESET FOR WRITES
//#####
    S9: begin
        short_timer <= 0;
        state <= S7;
    end

//#####
//S10: SHORT_TIMER RESET FOR READS
//#####
    S10: begin
        short_timer <= 0;
        state <= S1;
    end

//#####
//S11: INIT FOR DOWN(R,W0)
//#####
    S11: begin
        statecount <= 2;
        DATA <= 8'b01010101;
        ADDR <= 15'b1111111111111111;
        state <= S10; //reset for read
    end

//#####
//S16: INIT FOR UP(R,W1)
//#####
    S16: begin
        statecount <= 1;
        DATA <= 8'b10101010;
        ADDR <= 15'b0000000000000000;
        state <= S10; //reset for read
    end

endcase

    end //end IF RESET ELSE

    end // end always
endmodule //END MEMCONTROL MODULE

```

APPENDIX D.
VERILOG CODE AND USER CONSTRAINT FILES (CD-ROM)

Verilog code and the corresponding user constraint file for each memory type and test type are included on CD-ROM in ISO-9660 format.

(If included, CD is to be labeled as follows):

Vikram M. N. Rao

An FPGA Based Testbench for Reliability and Endurance Characterization of Nonvolatile Memory

M.S. Thesis, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2002

Appendix D: Verilog Code and User Constraint Files
